Aula 4 - Data Management, Notifications and Fragments

Summary:

- Data Management
- Notifications
- Fragments

4.1. Data Management

In Android you might use the following options for data management:

- Internal file storage: Store app-private files on the device file system.
- External file storage: Store files on the shared external file system. This is usually for shared user files, such as photos.
- Shared preferences: Store private primitive data in key-value pairs.
- Databases: Store structured data in a private database.

Create a new project name DM with the following Layout:

```
<TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_marginStart="50dp"
        android:layout_marginTop="67dp"
        android:layout_marginEnd="50dp"
        android:layout_marginBottom="46dp"
        android:padding="5dp"
        android:text="Read and Write Text from/to a File"
        android:textSize="28sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/editText2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/textMessage"
        android:layout_width="374dp"
        android:layout_height="0dp"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="1dp"
        android:minLines="5"
        app:layout_constraintBottom_toTopOf="@+id/button1"
        app:layout_constraintEnd_toStartOf="@+id/button4"
        app:layout_constraintStart_toStartOf="@+id/button4"
        app:layout_constraintTop_toBottomOf="@+id/textView2">

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/button2"
        android:layout_alignEnd="@+id/editText2"
        android:layout_alignRight="@+id/editText2"
        android:layout_marginStart="5dp"
        android:layout_marginEnd="7dp"
        android:layout_marginBottom="16dp"
        android:text="Write Text into File"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toStartOf="@+id/button4"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText2" />

    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_alignStart="@+id/editText2"
        android:layout_alignLeft="@+id/editText2"
        android:layout_centerVertical="true"
        android:layout_marginStart="5dp"
        android:layout_marginEnd="7dp"
        android:layout_marginBottom="33dp"
        android:text="Read Text From file"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button1" />

    <Button
        android:id="@+id/button4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/button2"
        android:layout_alignEnd="@+id/editText2"
        android:layout_alignRight="@+id/editText2"
        android:layout_marginEnd="12dp"
        android:text="Write Prefs"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/button1"
        app:layout_constraintTop_toBottomOf="@+id/editText2" />

    <Button
        android:id="@+id/button5"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_alignStart="@+id/editText2"
        android:layout_alignLeft="@+id/editText2"
        android:layout_centerVertical="true"
        android:layout_marginEnd="12dp"
        android:text="Read Text Prefs"
        app:layout_constraintBaseline_toBaselineOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/button2" />
```

Then, in the Android manifest.xml file add the following line to provide permissions to write.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Finally, modify the project MainActivity to read and write the text from the file "mytextfile.txt".

Example of file writing:

```
static final int READ_BLOCK_SIZE = 100;

try {
    FileOutputStream fileout=openFileOutput("mytextfile.txt", MODE_PRIVATE);
    OutputStreamWriter outputWriter=new OutputStreamWriter(fileout);
    outputWriter.write(textmsg.getText().toString());
    outputWriter.close();

    Toast.makeText(getBaseContext(), "File saved successfully!",
    Toast.LENGTH_SHORT).show();
} catch (Exception e) {
    e.printStackTrace();
}
```

Example of file reading:

```
try {
    FileInputStream fileIn=openFileInput("mytextfile.txt");
    InputStreamReader InputRead= new InputStreamReader(fileIn);

    char[] inputBuffer= new char[READ_BLOCK_SIZE];
    String s="";
    int charRead;

    while ((charRead=InputRead.read(inputBuffer))>0) {
        String readstring=String.copyValueOf(inputBuffer,0,charRead);
        s +=readstring;
    }
    InputRead.close();
    textmsg.setText(s);
} catch (Exception e) {
    e.printStackTrace();
}
```

Modify the project to perform the same operation using Shared Preferences.

<u>Example of file Shared Preferences Reading/writing:</u>

Writing:

```
SharedPreferences preferences =
        PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
SharedPreferences.Editor editor = preferences.edit();
editor.putInt("score", score );
editor.apply();
```

Reading:

```
SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
int score = preferences.getInt("score", 0);
textView.setText("Score " + score);
```

More info:
https://developer.android.com/guide/topics/data/data-storage
https://developer.android.com/reference/android/content/SharedPreferences.html
https://developer.android.com/training/basics/data-storage/shared-preferences.html

4.2. Notifications

A notification in its most basic and compact form displays an icon, a title, and a small amount of content text. Modify the previous example to show a notification on reading/writing information to a file.

A notification's content needs to be set through a NotificationCompat.Builder.

```
NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(MainActivity.this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Much longer text that cannot fit one line...")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

It is also necessary to create an intent. The method setContentIntent associate the pendingIntent to the notification.

```
Intent resultIntent = new Intent(this, MainActivity.class);
PendingIntent resulPendingIntent = PendingIntent.getActivity(this, 0,
resultIntent, PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resulPendingIntent);
```

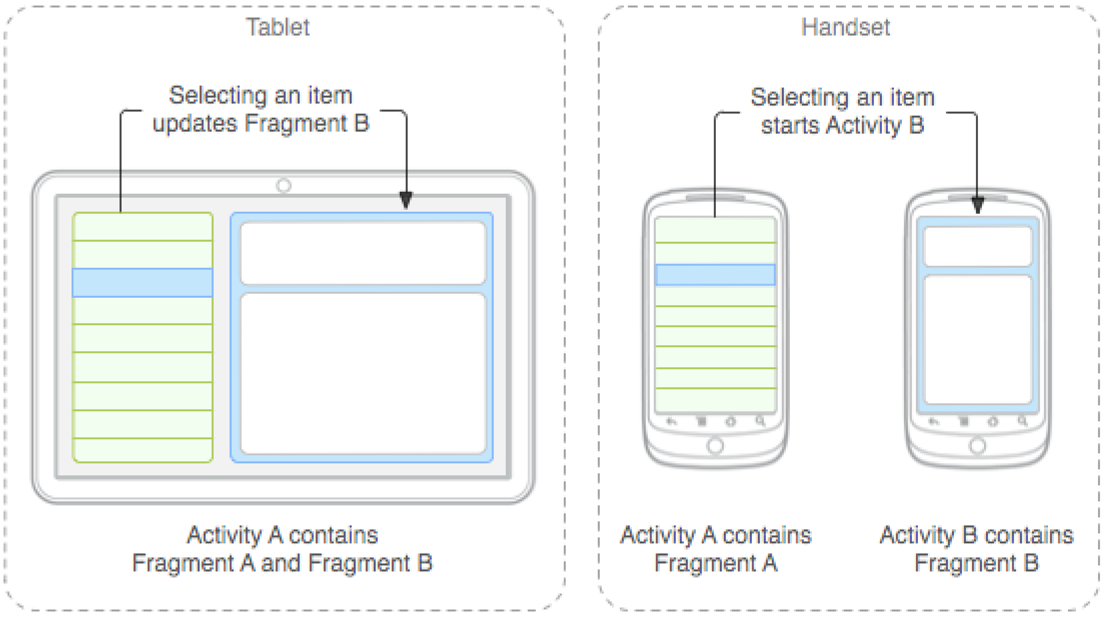
Finally, the method notify from the NotificationManager is responsible to pass the notification to the system and return its ID for posterior use:

```
int notificationId = 1;
NotificationManager mNotificationManager = NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(notificationId, mBuilder.build());
```

More info:
Building a notification:
https://developer.android.com/training/notify-user/build-notification.html
User Interface » Notifications:
https://developer.android.com/guide/topics/ui/notifiers/notifications.html

4.3. Fragments

Fragment represents a behaviour or a portion of user interface in a FragmentActivity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity.


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

Fragments should be reusable, hosted in an activity and define part of an interface. There are several sub-classes of fragments:

- DialogFragment; ListFragment; PreferenceFragmentCompat, etc.

A fragment, like an activity, has an XML layout file and a Java class that represents the Fragment controller.

We are going to modify the previous example to show the output of the reading in a fragment.

Create a blank fragment (called FragmentRead) with the following layout (uncheck the options include fragment factory methods and include interface callbacks).

Modify the fragment_read_layout to include only a TextView as follow (you may update the height):

```
<TextView
    android:id="@+id/textViewFrag"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:text="I'm a fragment!"
    android:textAlignment="center"
    android:textColor="@android:color/white"
    android:background="@android:color/holo_green_dark"
    android:layout_height="260dp"
    android:textSize="24sp" />
```

<u>Static declaration of the fragment:</u>

You may now add the fragment directly in your activity with the following code:

```
    <fragment
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/fragment1"
```